# Dokumentacja Projektu: Zarządzanie Połączeniami Lotniczymi - UPDATE

Krzysztof Czerwiński

11.12.2024r.

## 1 Funkcjonalności

Aplikacja obsługuje następujące funkcjonalności:

- Wyszukiwanie lotnisk według kraju i innych parametrów.
- Wyszukiwanie bezpośrednich połączeń lotniczych między lotniskami.
- Wyszukiwanie niebezpośrednich połączeń lotniczych.
- Dodawanie nowych lotnisk do bazy danych.
- Dodawanie nowych połączeń lotniczych między lotniskami.
- Usuwanie lotnisk z możliwością usunięcia wszystkich powiązanych połączeń.
- Usuwanie konkretnych połączeń lotniczych.
- Wizualizacja połączeń lotniczych dla wybranego lotniska z użyciem interaktywnego graficznego interfejsu.

## 2 Aplikacja Frontendowa

Frontend aplikacji został stworzony przy użyciu React i TypeScript. Aplikacja umożliwia użytkownikowi:

- Wyszukiwanie i filtrowanie lotnisk oraz połączeń.
- Dodawanie i usuwanie lotnisk oraz połączeń z bazy danych.
- Wizualizację połączeń lotniczych w postaci grafu z użyciem biblioteki `vis-network`.

Przy każdej operacji dane są przesyłane do API.

# 3 Aplikacja Backendowa

Backend został zaimplementowany w Pythonie z użyciem frameworka Flask. Obsługuje on:

- Endpointy REST API dla wyszukiwania, dodawania i usuwania lotnisk oraz połączeń.
- Połączenie z bazą danych Neo4j, która przechowuje dane o lotniskach i połączeniach w formie grafu.
- Wyszukiwanie połączeń między węzłami.

# 4 Baza danych

Dane w bazie danych to dane rzeczywiste(z 2009 roku), pochodzące ze strony `https://openflights.org/data.php`. Zostały wyeksportowane do bazy przy pomocy stworzonych skryptów z plików CSV.

# 5 Dokeryzacja

Aplikacja została zkonteneryzowana przy użyciu Docker:

- Stworzono osobny Dockerfile dla aplikacji frontendowej, która po zbudowaniu plików statycznych (`npm run build`) umieszcza je w folderze `static/react` w backendzie.
- Backend posiada Dockerfile obsługujący instalację zależności oraz uruchamianie aplikacji Flask.
- Przy użyciu Docker Compose można uruchomić obie aplikacje jako osobne usługi z odpowiednio skonfigurowanymi zmiennymi środowiskowymi.

# 6 Umieszczenie w Chmurze

Aplikacja została wdrożona w chmurze Microsoft Azure:

- Wykorzystano Azure App Service do hostowania zarówno backendu, jak i frontendowej aplikacji React.
- Obie aplikacje zostały umieszczone w obrazach Docker, które są automatycznie budowane i przesyłane do Azure Container Registry (ACR).
- Frontend i backend działają jako oddzielne usługi z Load Balancerem umożliwiającym skalowalność.
- Zmiennymi środowiskowymi skonfigurowano adresy API oraz porty.

# 7 Zmienne środowiskowe

Podczas wdrażania aplikacji należy skonfigurować następujące zmienne środowiskowe:

- `FLASK_APP_PORT` - Port, na którym działa backend.
- `NEO4J_URI` - Adres bazy danych Neo4j.
- `NEO4J_USER` - Nazwa użytkownika bazy danych Neo4j.
- `NEO4J_PASSWORD` - Hasło do bazy danych Neo4j.